ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
по курсу
«Data Science»

Докладчик: Матюнин Александр Александрович

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Цель и задача работы

Целью работы является изучение теоретических основ и методов решения поставленной задачи. Создать и разработать приложение, которое будет предсказывать ряд конечных свойств композиционных материалов основываясь на предобученных моделях.

Задачи данной работы:

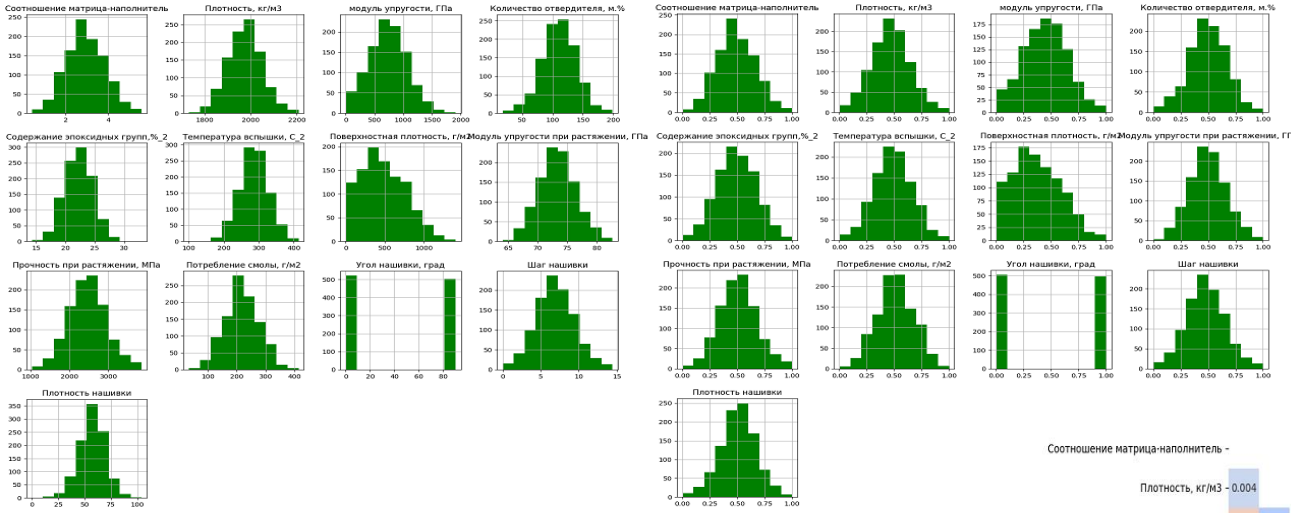
- Изучить теоретические основы и методы решения поставленной задачи.
- Провести разведочный анализ предложенных данных. Необходимо нарисовать гистограммы распределения каждой из переменной, диаграммы ящика с усами, попарные графики рассеяния точек. Необходимо также для каждой колонки получить среднее, медианное значение, провести анализ и исключение выбросов, проверить наличие пропусков.
- Провести предобработку данных
- Обучить нескольких моделей для прогноза
- Написать нейронную сеть
- Разработать приложение с графическим интерфейсом
- Создать репозиторий в GitHub / GitLab и разместить там код исследования

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Начало работы:

- Изучение теоретических основ, методов решения и практических составляющих
- Построение графиков
- Унификация стиля графиков
- Создание понятных переменных, для дальнейшего использования
- Создание функций для упрощения и ускорения написания кода

Гистограммы распределения данных до нормализации и после нормализации, в основном симметричные

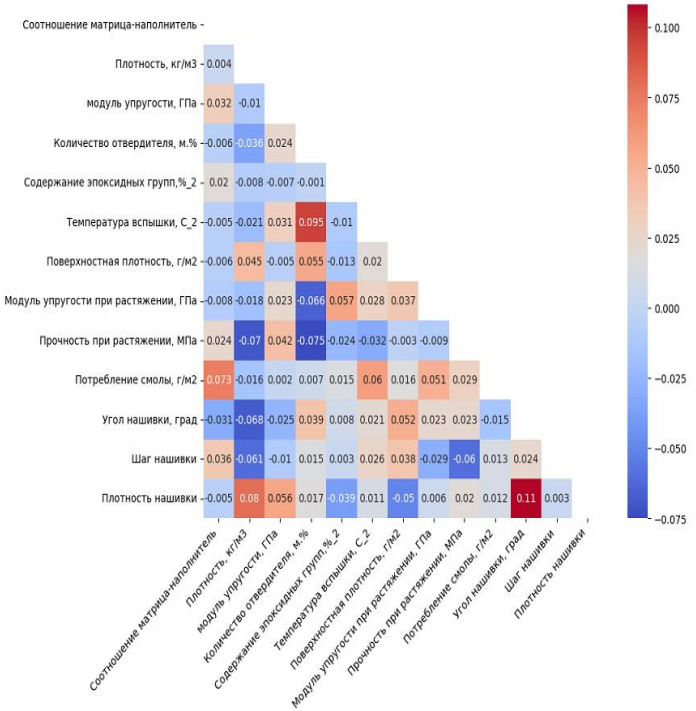



```python
# создаем переменную "corr_round" с округлением до 3 знаков после запятой
corr_round = df_nb.corr().round(3)
mask = np.triu(df_nb.corr())
f, ax = plt.subplots(figsize = (10, 8))
sns.heatmap(corr_round, mask = mask, annot = True, square = True, cmap = 'coolwarm')
plt.xticks(rotation = 45, ha='right')
plt.show()
```

```python
def mase(y_test, y_pred, y_train):
    e_t = y_test - y_pred
    scale = mean_absolute_error(y_train[1:], y_train[:-1])
    return np.mean(np.abs(e_t / scale))
```

```python
def mape(y_test, pred):
    y_test, pred = np.array(y_test), np.array(pred)
    mape = np.mean(np.abs((y_test - pred) / y_test))
    return mape
```

Пример функций (MASE и MAPE)



Тепловая карта корреляции

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Объединение файлов и разведочный анализ:

- Импорт необходимых библиотек и их версии

```
import numpy as np
import pandas as pd
import seaborn
import seaborn as sns
import matplotlib.pyplot as plt
import scipy
from scipy import stats
%matplotlib inline
import sklearn
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input, BatchNormalization, LeakyReLU, Activation, Dropout
import tensorflow as tf
import tensorflow
import keras
from tensorflow.keras import layers
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam


print(f'numpy ver: {np.__version__}')
print(f'pandas ver: {pd.__version__}')
print(f'seaborn ver: {sns.__version__}')
print(f'sklearn ver: {sklearn.__version__}')
print(f'scipy ver: {scipy.__version__}')
print(f'tensorflow ver: {tensorflow.__version__}')
print(f'keras ver: {keras.__version__}')
```

```
numpy ver: 1.24.3
pandas ver: 2.0.3
seaborn ver: 0.12.2
sklearn ver: 1.3.0
scipy ver: 1.11.2
tensorflow ver: 2.13.0
keras ver: 2.13.1
```

- Загрузка файлов
- Проверка размерности

```
# первую колонку (порядковый номер) используем как индекс
df_bp = pd.read_excel(r'C:\Users\Александр\Desktop\VKR_KOMPOZIT\Data\X_bp.xlsx', index_col = 'Unnamed: 0')
df_nup = pd.read_excel(r'C:\Users\Александр\Desktop\VKR_KOMPOZIT\Data\X_nup.xlsx', index_col = 'Unnamed: 0')
```

```
# смотрим количество сток в датасете(размерность строк)
print(f'Колличество строк в датасете X_bp.xlsx  {df_bp.shape[0]}')
print(f'Колличество строк в датасете X_nup.xlsx {df_nup.shape[0]}')
```

```
Колличество строк в датасете X_bp.xlsx  1023
Колличество строк в датасете X_nup.xlsx 1040
```

```
# смотрим колличество колонок в датасете(размерность колонок)
print(f'Колличество колонок в датасете X_bp.xlsx  {df_bp.shape[1]}')
print(f'Колличество колонок в датасете X_nup.xlsx {df_nup.shape[1]}')
```

```
Колличество колонок в датасете X_bp.xlsx  10
Колличество колонок в датасете X_nup.xlsx 3
```

- Объединение файлов по индексу, тип объединения INNER
- Посмотрим на данные объединенного датасета

```
# Эти два датасета имеют разный объем строк.
# Наша задача собрать исходные данные файлы в один, единый набор данных.
# По условию задачи объединяем их по типу INNER
df_nb = pd.merge(df_bp, df_nup, how="inner", left_on=None, right_on=None, left_index=True, right_index=True)
df_nb.head()
```

Python

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	1.857143	2030.0	738.736842	30.00	22.267857	100.000000	210.0	70.0	3000.0	220.0	0	4.0	57.0
1	1.857143	2030.0	738.736842	50.00	23.750000	284.615385	210.0	70.0	3000.0	220.0	0	4.0	60.0
2	1.857143	2030.0	738.736842	49.90	33.000000	284.615385	210.0	70.0	3000.0	220.0	0	4.0	70.0
3	1.857143	2030.0	738.736842	129.00	21.250000	300.000000	210.0	70.0	3000.0	220.0	0	5.0	47.0
4	2.771331	2030.0	753.000000	111.86	22.267857	284.615385	210.0	70.0	3000.0	220.0	0	5.0	57.0

```
# смотрим размерность обьединенного датафрейма
df_nb.shape
# Итоговый датасет имеет 13 столбцов и 1023 строки, 17 строк из таблицы X_nup было отброшено,т.е часть данных была удалена на начальном этапе исследования.
# При использовании "Inner" из результатов запроса будут удалены все строки, которым не нашлась соответствующая пара в другой таблице
```

Python

```
(1023, 13)
```

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Объединение файлов и разведочный анализ:

- Просмотр уникальных значений с помощью встроенной функции nunique

```
#Поиск уникальных значений с помощью функци
df_nb.nunique()
#Видим в основном общее число уникальных зн
# можно провести кодировку, но если в после
```

```
Соотношение матрица-наполнитель          1014
Плотность, кг/м3                         1013
модуль упругости, ГПа                    1020
Количество отвердителя, м.%              1005
Содержание эпоксидных групп,%_2          1004
Температура вспышки, С_2                 1003
Поверхностная плотность, г/м2            1004
Модуль упругости при растяжении, ГПа     1004
Прочность при растяжении, МПа            1004
Потребление смолы, г/м2                  1003
Угол нашивки, град                          2
Шаг нашивки                               989
Плотность нашивки                         988
dtype: int64
```

- Проверка на пропуски и дубликаты

```
# Проверим датасет на дубликаты
df_nb.duplicated().sum()
#Дубликатов нет
```

```
0
```

```
#Проверка на пропущенных значений
df_nb.isna().sum()
```

```
Соотношение матрица-наполнитель          0
Плотность, кг/м3                         0
модуль упругости, ГПа                    0
Количество отвердителя, м.%              0
Содержание эпоксидных групп,%_2          0
Температура вспышки, С_2                 0
Поверхностная плотность, г/м2            0
Модуль упругости при растяжении, ГПа     0
Прочность при растяжении, МПа            0
Потребление смолы, г/м2                  0
Угол нашивки, град                       0
Шаг нашивки                              0
Плотность нашивки                        0
dtype: int64
```

- Проверим типы данных в каждом столбце

```
# смотрим параметры обьединенного DataFrame
df_nb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1023 entries, 0 to 1022
Data columns (total 13 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Соотношение матрица-наполнитель       1023 non-null   float64
 1   Плотность, кг/м3                      1023 non-null   float64
 2   модуль упругости, ГПа                 1023 non-null   float64
 3   Количество отвердителя, м.%           1023 non-null   float64
 4   Содержание эпоксидных групп,%_2       1023 non-null   float64
 5   Температура вспышки, С_2              1023 non-null   float64
 6   Поверхностная плотность, г/м2         1023 non-null   float64
 7   Модуль упругости при растяжении, ГПа  1023 non-null   float64
 8   Прочность при растяжении, МПа         1023 non-null   float64
 9   Потребление смолы, г/м2               1023 non-null   float64
 10  Угол нашивки, град                    1023 non-null   int64
 11  Шаг нашивки                           1023 non-null   float64
 12  Плотность нашивки                     1023 non-null   float64
dtypes: float64(12), int64(1)
memory usage: 111.9 KB
```

- Описательная статистика

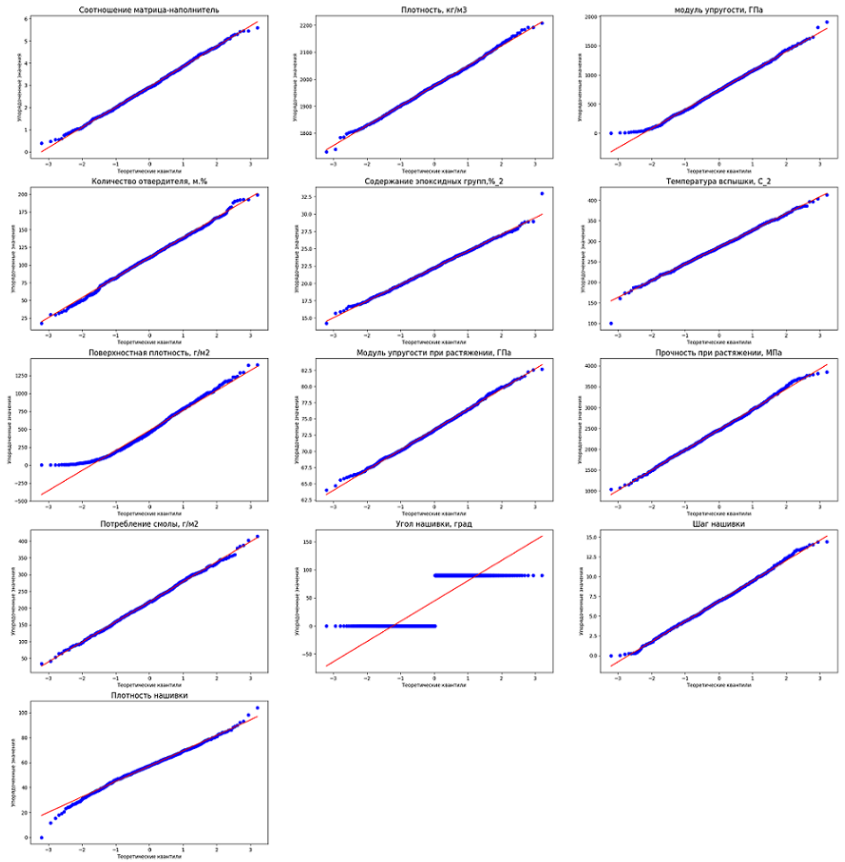
```
# рассчитаем основные статистические показатели
df_nb.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1023.0	2.930366	0.913222	0.389403	2.317887	2.906878	3.552660	5.591742
Плотность, кг/м3	1023.0	1975.734888	73.729231	1731.764635	1924.155467	1977.621657	2021.374375	2207.773481
модуль упругости, ГПа	1023.0	739.923233	330.231581	2.436909	500.047452	739.664328	961.812526	1911.536477
Количество отвердителя, м.%	1023.0	110.570769	28.295911	17.740275	92.443497	110.564840	129.730366	198.953207
Содержание эпоксидных групп,%_2	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.961934	33.000000
Температура вспышки, С_2	1023.0	285.882151	40.943260	100.000000	259.066528	285.896812	313.002106	413.273418
Поверхностная плотность, г/м2	1023.0	482.731833	281.314690	0.603740	266.816645	451.864365	693.225017	1399.542362
Модуль упругости при растяжении, ГПа	1023.0	73.328571	3.118983	64.054061	71.245018	73.268805	75.356612	82.682051
Прочность при растяжении, МПа	1023.0	2466.922843	485.628006	1036.856605	2135.850448	2459.524526	2767.193119	3848.436732
Потребление смолы, г/м2	1023.0	218.423144	59.735931	33.803026	179.627520	219.198882	257.481724	414.590628
Угол нашивки, град	1023.0	44.252199	45.015793	0.000000	0.000000	0.000000	90.000000	90.000000
Шаг нашивки	1023.0	6.899222	2.563467	0.000000	5.080033	6.916144	8.586293	14.440522
Плотность нашивки	1023.0	57.153929	12.350969	0.000000	49.799212	57.341920	64.944961	103.988901

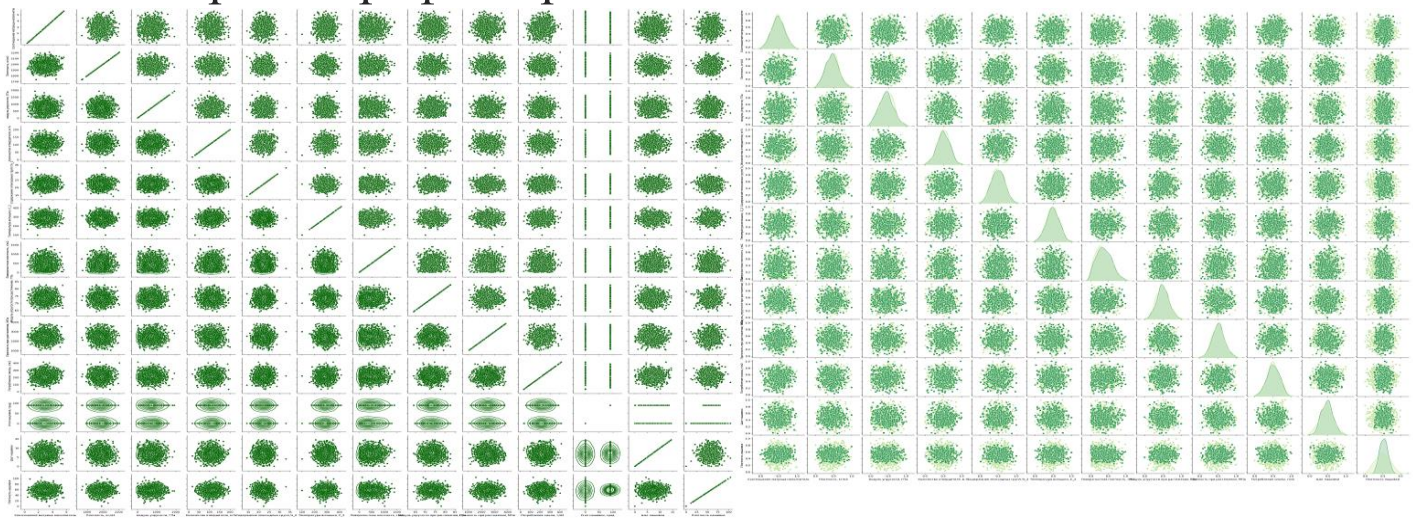
ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Визуализация «сырых» данных:

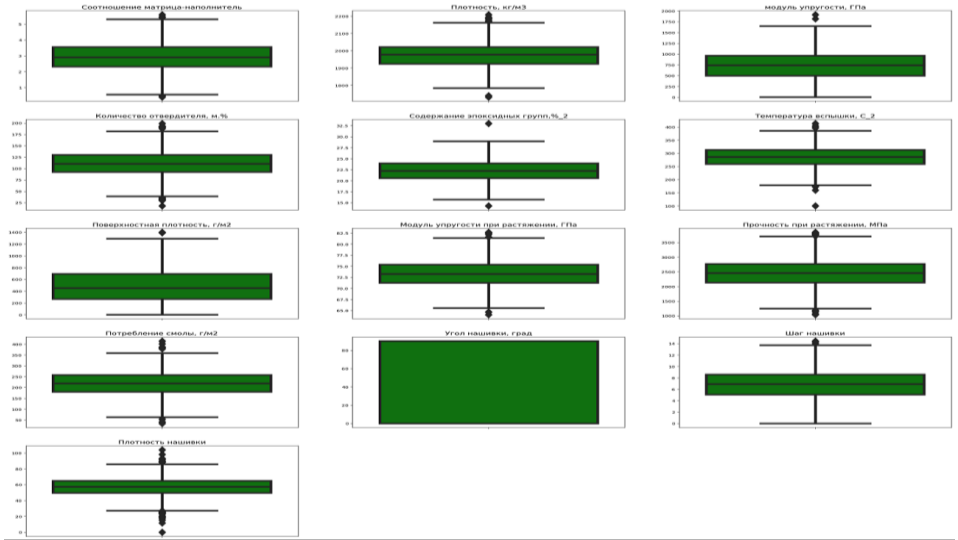
- График вероятности для распределения данных



- Попарные графики рассеяния точек



- Диаграммы Box Plot «ящики с усами» для определения выбросов



ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Предобработка данных:

- Подсчет количества значений методом 3 сигм и методом межквартильных расстояний
- Исключим выбросы методом 3 сигм
- Посмотрим описание

```
# методом 3-х сигм
count_3s = 0
for column in df_nb:
    d = df_nb.loc[:, [column]]
    zscore = (df_nb[column] - df_nb[column].mean()) / df_nb[column].std()
    d['3s'] = zscore.abs() > 3
    count_3s += d['3s'].sum()

print('Метод 3-х сигм, выбросов:', count_3s)
```

```
Метод 3-х сигм, выбросов: 24
```

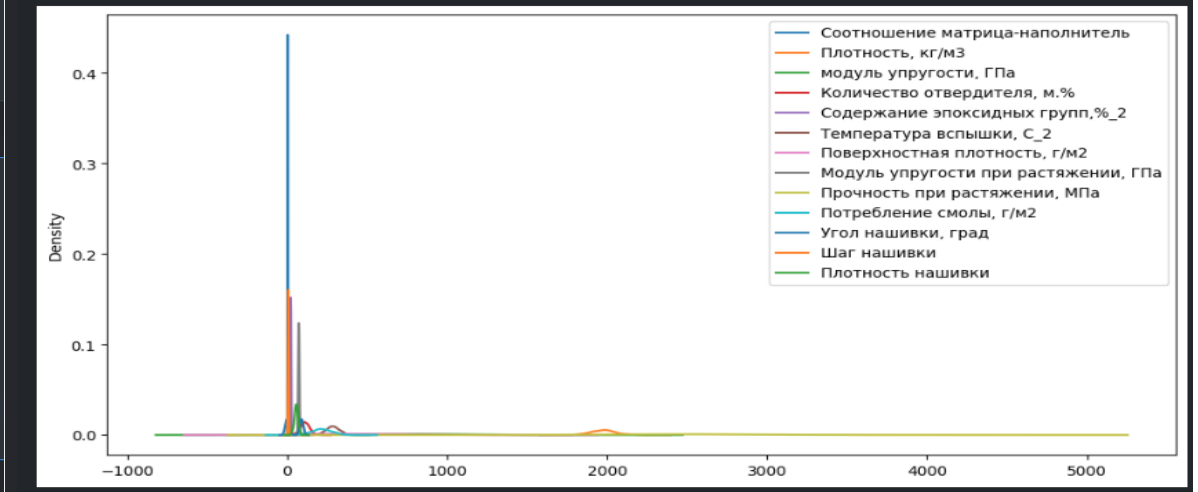
```
# метод межквартильных расстояний
count_iq = 0
for column in df_nb:
    d = df_nb.loc[:, [column]]
    q1 = np.quantile(df_nb[column], 0.25)
    q3 = np.quantile(df_nb[column], 0.75)
    iqr = q3 - q1
    lower = q1 - 1.5 * iqr
    upper = q3 + 1.5 * iqr
    d['iq'] = (df_nb[column] <= lower) | (df_nb[column] >= upper)
    count_iq += d['iq'].sum()

print('Метод межквартильных расстояний, выбросов:', count_iq)
```

```
Метод межквартильных расстояний, выбросов: 93
```

```
fig, ax = plt.subplots(figsize = (12, 6))
df_nb_clear.plot(kind = 'kde', ax = ax)
```

```
<Axes: ylabel='Density'>
```




```
# Удаляем выбросы методом 3-х сигм
outliers = pd.DataFrame(index=df_nb.index)
for column in df_nb:
    zscore = (df_nb[column] - df_nb[column].mean()) / df_nb[column].std()
    outliers[column] = (zscore.abs() > 3)
df_nb_clear = df_nb[outliers.sum(axis=1)==0]
df_nb_clear.shape
```

```
(1000, 13)
```

```
minmax_and_mean50_cl = df_nb_clear.describe()
minmax_and_mean50_cl.loc[['min', 'max', 'mean', '50%']].T
```

	min	max	mean	50%
Соотношение матрица-наполнитель	0.389403	5.591742	2.936299	2.908811
Плотность, кг/м3	1784.482245	2192.738783	1975.402478	1977.321002
модуль упругости, ГПа	2.436909	1649.415706	738.675486	741.148111
Количество отвердителя, м.%	29.956150	192.851702	110.821904	110.652620
Содержание эпоксидных групп,%_2	15.695894	28.955094	22.235549	22.221462
Температура вспышки, С_2	173.484920	403.652861	285.957299	285.853960
Поверхностная плотность, г/м2	0.603740	1291.340115	479.855825	450.869535
Модуль упругости при растяжении, ГПа	64.054061	82.682051	73.318178	73.230375
Прочность при растяжении, МПа	1036.856605	3848.436732	2464.864198	2456.394188
Потребление смолы, г/м2	41.048278	386.903431	218.254011	218.697660
Угол нашивки, град	0.000000	90.000000	44.640000	0.000000
Шаг нашивки	0.037639	14.440522	6.910600	6.922196
Плотность нашивки	20.571633	92.963492	57.276293	57.471971

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Предобработка данных:

- Нормализация данных MinMaxScaler()

Приводим данные к масштабированию с помощью MinMaxScaler

```
scaler = MinMaxScaler()
df_min_max = pd.DataFrame(scaler.fit_transform(df_nb_clear), columns=column_names)
```

Python

```
df_min_max.head()
```

Python

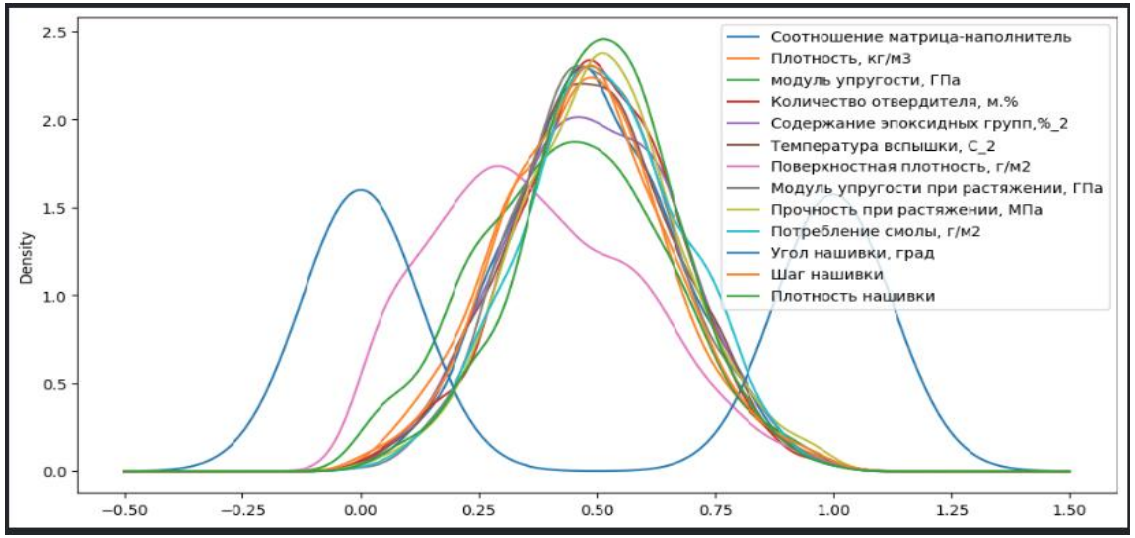
	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	0.282131	0.601381	0.447061	0.123047	0.607435	0.482823	0.16223	0.319194	0.698235	0.517418	0.0	0.275109	0.544652
1	0.282131	0.601381	0.447061	0.608021	0.418887	0.549664	0.16223	0.319194	0.698235	0.517418	0.0	0.344539	0.365074
2	0.457857	0.601381	0.455721	0.502800	0.495653	0.482823	0.16223	0.319194	0.698235	0.517418	0.0	0.344539	0.503211
3	0.457201	0.527898	0.452685	0.502800	0.495653	0.482823	0.16223	0.319194	0.698235	0.517418	0.0	0.344539	0.544652
4	0.419084	0.307448	0.488508	0.502800	0.495653	0.482823	0.16223	0.319194	0.698235	0.517418	0.0	0.344539	0.682789

- Выведение описательной статистики после нормализации

```
df_min_max.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1000.0	0.489568	0.174687	0.0	0.370964	0.484284	0.608289	1.0
Плотность, кг/м3	1000.0	0.467648	0.178696	0.0	0.340831	0.472347	0.579727	1.0
модуль упругости, ГПа	1000.0	0.447024	0.198876	0.0	0.302576	0.448525	0.582408	1.0
Количество отвердителя, м.%	1000.0	0.496427	0.171089	0.0	0.384097	0.495388	0.613258	1.0
Содержание эпоксидных групп,%_2	1000.0	0.493216	0.179818	0.0	0.368597	0.492154	0.624396	1.0
Температура вспышки, С_2	1000.0	0.488654	0.174792	0.0	0.371985	0.488205	0.606271	1.0
Поверхностная плотность, г/м2	1000.0	0.371301	0.215155	0.0	0.206374	0.348844	0.535295	1.0
Модуль упругости при растяжении, ГПа	1000.0	0.497322	0.167158	0.0	0.386234	0.492609	0.605138	1.0
Прочность при растяжении, МПа	1000.0	0.507902	0.172506	0.0	0.390414	0.504890	0.612932	1.0
Потребление смолы, г/м2	1000.0	0.512370	0.170432	0.0	0.401220	0.513653	0.625772	1.0
Угол нашивки, град	1000.0	0.496000	0.500234	0.0	0.000000	0.000000	1.000000	1.0
Шаг нашивки	1000.0	0.477193	0.177586	0.0	0.351886	0.477999	0.593714	1.0
Плотность нашивки	1000.0	0.507027	0.163634	0.0	0.405037	0.509730	0.612766	1.0

- Построение графика плотности распределения данных



ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Разработка и обучение моделей для прогноза упругости при растяжении, ГПа

- Построение моделей
- Обучение моделей
- Вычисление ошибок
- Поиск гиперпараметров методом GridSearchCV с перекрестной проверкой с количеством блоков 10
- Подставление оптимальных параметров

```
X_uprygost = df_min_max.drop(['Модуль упругости при растяжении, ГПа'], axis=1)
X_prochnost = df_min_max.drop(['Прочность при растяжении, МПа'], axis=1)
y_uprygost = df_min_max['Модуль упругости при растяжении, ГПа']
y_prochnost = df_min_max['Прочность при растяжении, МПа']

X_uprygost_train, X_uprygost_test, y_uprygost_train, y_uprygost_test = train_test_split(X_uprygost, y_uprygost, test_size = 0.3, shuffle = True)
X_prochnost_train, X_prochnost_test, y_prochnost_train, y_prochnost_test = train_test_split(X_prochnost, y_prochnost, test_size = 0.3, shuffle = True)

print('X_train: для Модуль упругости при растяжении, ГПа', X_uprygost_train.shape, 'y_train: для Модуль упругости при растяжении, ГПа', y_uprygost_train.shape)
print('X_test: для Модуль упругости при растяжении, ГПа', X_uprygost_test.shape, 'y_test: для Модуль упругости при растяжении, ГПа', y_uprygost_test.shape)
print('X_train: для Прочность при растяжении, МПа, ГПа', X_prochnost_train.shape, 'y_train: для Прочность при растяжении, МПа', y_prochnost_train.shape)
print('X_test: для Прочность при растяжении, МПа', X_prochnost_test.shape, 'y_test: для Прочность при растяжении, МПа', y_prochnost_test.shape)
```

```
X_train: для Модуль упругости при растяжении, ГПа (700, 12) y_train: для Модуль упругости при растяжении, ГПа (700,)
X_test: для Модуль упругости при растяжении, ГПа (300, 12) y_test: для Модуль упругости при растяжении, ГПа (300,)
X_train: для Прочность при растяжении, МПа, ГПа (700, 12) y_train: для Прочность при растяжении, МПа (700,)
X_test: для Прочность при растяжении, МПа (300, 12) y_test: для Прочность при растяжении, МПа (300,)
```

```
knn_uprygost = KNeighborsRegressor()
param_grid = {
    "n_neighbors":  range(1, 301, 2),
    "weights": ['uniform', 'distance'],
    "algorithm": ['auto', 'ball_tree', 'kd_tree', 'brute'],
    "p": range(1, 5, 2)
}
grid_search_knn_uprygost = GridSearchCV(knn_uprygost, param_grid, cv=10, verbose=1, scoring = 'neg_mean_squared_error')
grid_search_knn_uprygost.fit(X_uprygost_train, y_uprygost_train)
model_knn_uprygost = grid_search_knn_uprygost.best_estimator_
model_knn_uprygost.fit(X_uprygost_train, y_uprygost_train)
knn_uprygost_pred = model_knn_uprygost.predict(X_uprygost_test)
knn_uprygost_Bscore = grid_search_knn_uprygost.best_score_
print("Best params", grid_search_knn_uprygost.best_params_)
print("Best score", knn_uprygost_Bscore)
```

```
Fitting 10 folds for each of 2400 candidates, totalling 24000 fits
Best params {'algorithm': 'auto', 'n_neighbors': 73, 'p': 3, 'weights': 'uniform'}
Best score -0.0271448320617629 9
```

- Датафрейм с ошибками моделей

```
models_uprygost.T
```

	Модуль Упругости при растяжении	Модуль Упругости при растяжении	Модуль Упругости при растяжении	Модуля Упругости при растяжении	Модуль Упругости при растяжении	Модуль Упругости при растяжении	Модуль Упругости при растяжении
Model	KNeighborsRegressor_uprygost	LinearRegression_uprygost	SGDRegressor_uprygost	RandomForestRegressor_uprygost	GradientBoostingRegressor_uprygost	MLPRegressor_uprygost	DecisionTreeRegressor_uprygost
MASE	0.744	0.744	0.749	0.747	0.754	0.746	0.755
MAPE	0.339	0.339	0.349	0.34	0.34	0.34	0.343
MSE	0.029	0.029	0.029	0.029	0.03	0.029	0.03
R2 score	-0.013	-0.013	-0.002	-0.016	-0.035	-0.018	-0.035
Best score	-0.027145	-0.039372	-0.076482	-0.040744	-0.062736	-0.020307	-0.025649

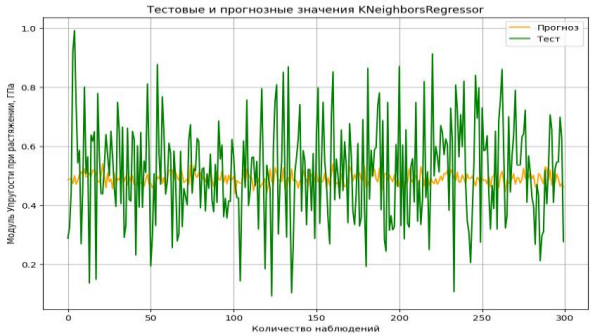
```
models_prochnost.T
```

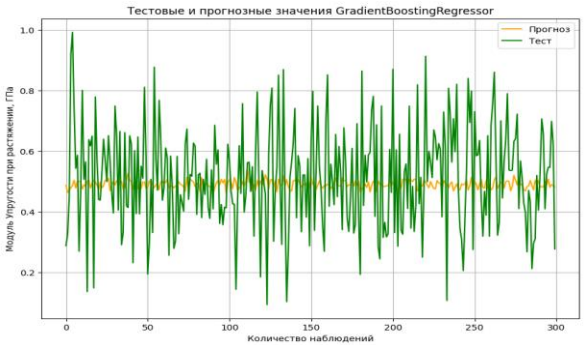
	Модуль Прочность при растяжении	Модуля Прочность при растяжении	Модуль Прочность при растяжении	Модуль Прочность при растяжении	Модуль Прочность при растяжении	Модуль Прочности при растяжении	Модуль Прочности при растяжении
Model	KNeighborsRegressor_prochnost	LinearRegression_prochnost	SGDRegressor_prochnost	RandomForestRegressor_prochnost	GradientBoostingRegressor_prochnost	MLPRegressor_prochnost	DecisionTreeRegressor_prochnost
MASE	0.689	0.697	0.709	0.692	0.733	0.69	0.701
MAPE	inf	inf	inf	inf	inf	inf	inf
MSE	0.03	0.03	0.03	0.03	0.032	0.03	0.03
R2 score	-0.008	-0.006	-0.037	-0.006	-0.083	-0.013	-0.031
Best score	-0.029803	-0.025615	-0.075167	-0.009232	-0.032024	-0.006573	-0.01405

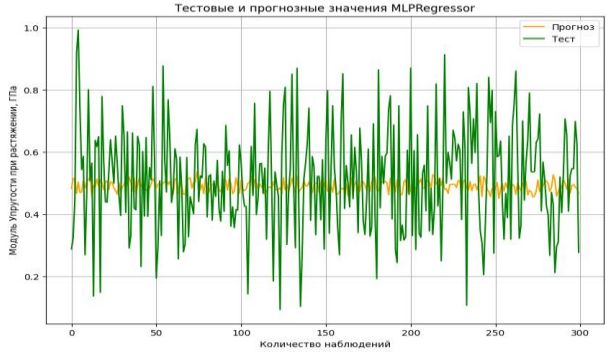
ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

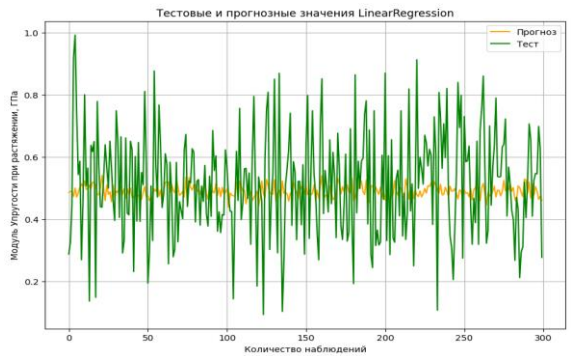
Разработка и обучение моделей для прогноза упругости при растяжении, ГПа

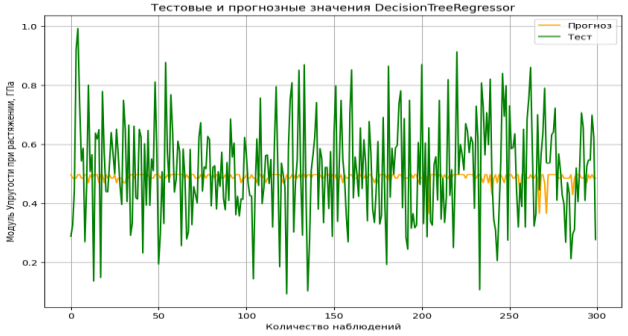
- Графики тестовых и прогнозных значений для разных методов

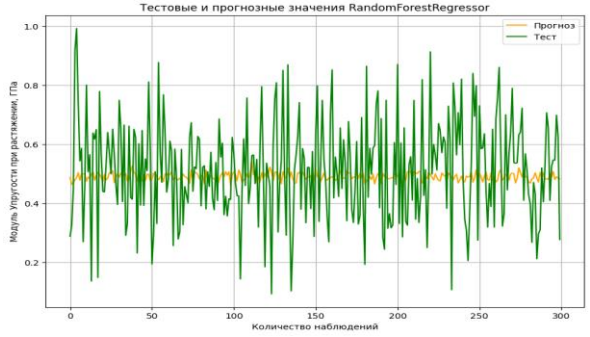


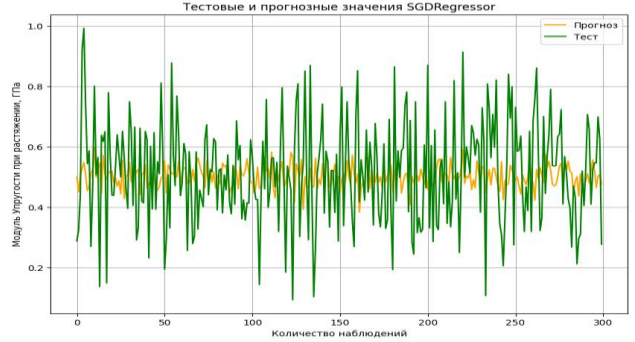




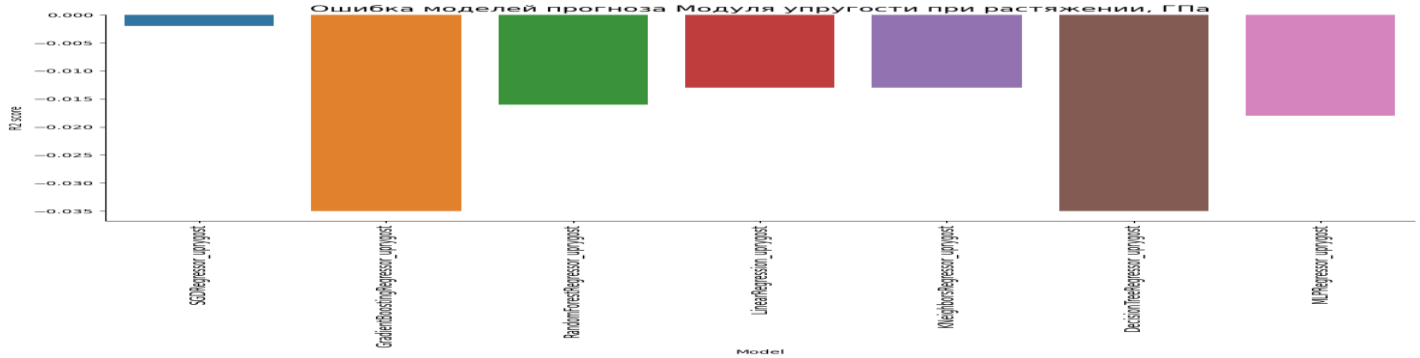








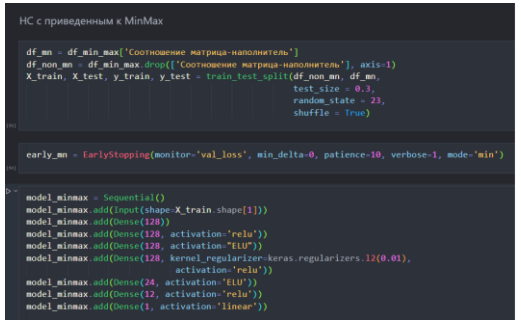
- График ошибок

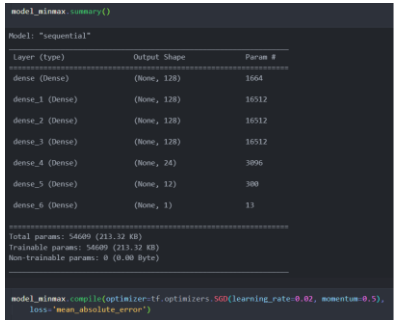


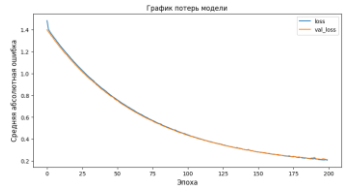
ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

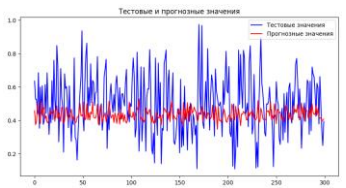
Нейронная сеть для соотношения «матрица-наполнитель»:

- Формирование входов и выходов для модели
- Разбивка на обучающую и тестовую выборки
- Построение модели и оценка результатов
- Обучение нейросети
- Сохранение модели



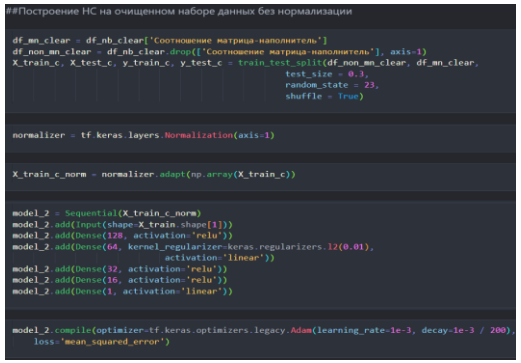


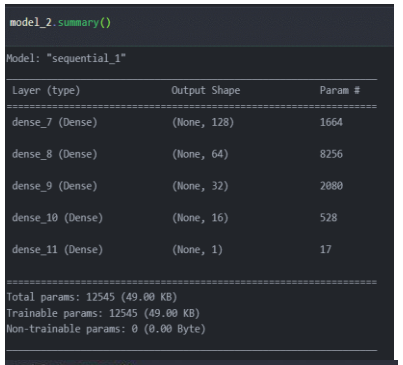


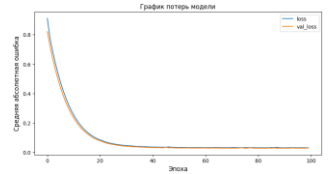


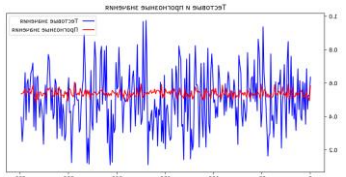

```
model_minmax.save('saved_model/model_minmax.h5')
```

```
C:\Users\Александр\AppData\Roaming\Python\Python311\si
instead the native Keras format, e.g. `model.save('my_
  saving_api.save_model(
```



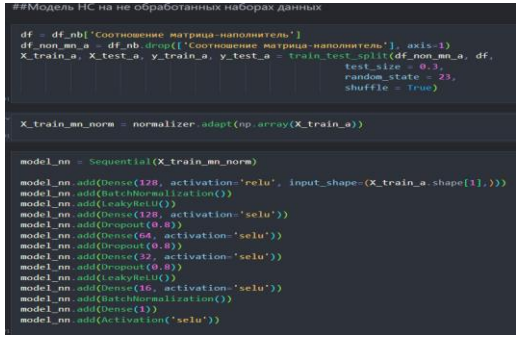


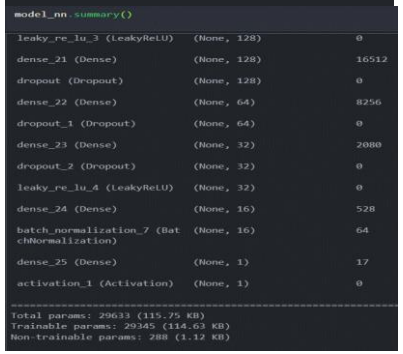


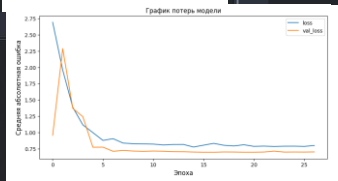


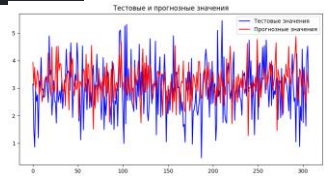
```
model_2.save('saved_model/model_clear.h5')
```

```
C:\Users\Александр\AppData\Roaming\Python\Pyt
instead the native Keras format, e.g. `model.
  saving_api.save_model(
```









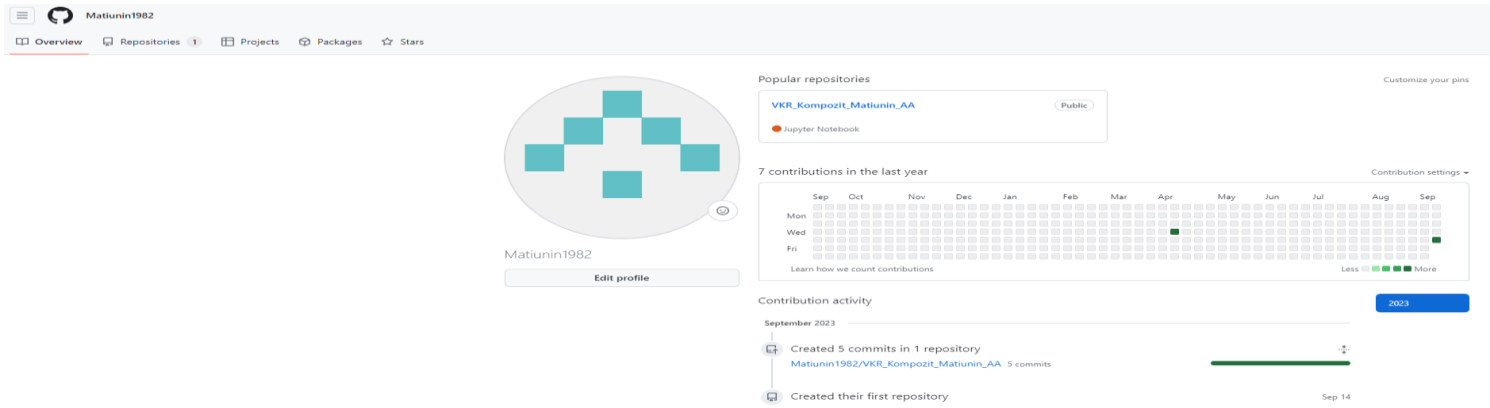

```
model_nn.save('saved_model/model_not_clear.h5')
```

```
C:\Users\Александр\AppData\Roaming\Python\Python311
instead the native Keras format, e.g. `model.save('
  saving_api.save_model(
```

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Приложение:

- Сохранение модели для разработки веб-приложения для прогнозирования соотношения «модуль упругости при растяжении» в фреймворке Flask
- При запуске приложения, пользователь переходит на: http://127.0.0.1:5000
- В открывшемся окне пользователю необходимо ввести в соответствующие ячейки требуемые значения и нажать на кнопку «Рассчитать».
- На выходе пользователь получает результат прогноза для значения параметра «Соотношение «матрица – наполнитель»».
- Приложение успешно
- Репозиторий на github.com
- https://github.com/Matiunin1982/VKR_Kompozit_Matiunin_AA/tree/main



ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Трудности и ошибки

- Опечатки, описки, пропуски скобок.
- Составление функций
- Когда что то не получалось или ломалась часть кода приходилось долго просматривать различные источники и много раз пробовать
- Самой большой трудностью было выкладывание на гитхаб, часть кода писал в гугл колабе, но приходилось писать и в VSCode, вот с ним то и пришлось помучиться, когда нужно было выложить всю работу.
- Пробовал работать в PyCharm, но возможно из за того, что уже привык к VSCode писалось намного медленнее, хотя все приходит с опытом.

Заключение

- Использованные при разработке моделей подходы не позволили получить некоторое количество достоверных прогнозов.
- Примененные модели регрессии не показали высокой эффективности в прогнозировании свойств композитов.
- Невозможно определить из свойств материалов соотношение «матрица – наполнитель».
- Текущим набором алгоритмов текущая задача эффективно не решается.

Спасибо за внимание!

ЦЕНТР
ДОПОЛНИТЕЛЬНОГО
ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

do.bmstu.ru